View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Caltech Authors

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

# Advanced load-testing techniques for a science archive

Mark Legassie, Lee Bennett, Susan Comeau, Suzanne Dodd

Mark Legassie, Lee Bennett, Susan Comeau, Suzanne Dodd, "Advanced load-testing techniques for a science archive," Proc. SPIE 7016, Observatory Operations: Strategies, Processes, and Systems II, 70161C (12 July 2008); doi: 10.1117/12.788045

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2008, Marseille, France

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018 Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

# Advanced Load Testing Techniques For a Science Archive

Mark Legassie*[a], Lee Bennett[b], Susan Comeau[a], Suzanne Dodd[b]
[a]Raytheon Information Solutions, 299 N Euclid Ave, Suite 500, Pasadena, CA 91101 USA;
[b]Spitzer Science Center, California Institute of Technology, M/S 314-6, 1200 E California Blvd, Pasadena, CA 91125 USA

## ABSTRACT

Performance goals for data archive systems need to be established early in the design process to ensure stability and acceptable response throughput. Load testing is one technique used to measure the progress towards these performance goals. Providing resources for load-test planning is critical, and this planning must include feasibility studies, tool analyses, and generation of an overall load-test strategy. This strategy is much different for science data archives than other systems, including commercial websites and high-volume data centers. This paper will provide an overview of the load testing performed on the Spitzer Space Telescope's science archive, which is part of Science Operations System at the Spitzer Science Center (SSC). Methods used for planning and conducting SSC load tests will be presented, and advanced load-testing techniques will be provided to address runtime issues and enhance verification results. This work was performed at the California Institute of Technology under contract to the National Aeronautics and Space Administration.

**Keywords:** Load Testing, Performance Testing, Science Archive, Spitzer Space Telescope

## 1. INTRODUCTION

### 1.1 Load Testing Overview

Load testing is the simulation of real-world loads on computer systems. A load test is performed to assess how the computer system will work under actual conditions of use. The test uses a number of virtual users, which are programs (scripts) that act like a real user would when making requests to the system. During load tests, a considerable number of virtual users can be run on one computer, making the testing environment cost-effective and easier to maintain.

Database load testing has been common, and now Internet load testing is becoming a fast-growing field. Although the internet and multi-tier environments have been around for many years, large-scale uses of internet load testing were not practical until recent advancements in load testing tools and techniques became available.

The overall goal when analyzing performance is to ensure that the computer system is operating with optimum stability, responsiveness, scalability, and throughput:

- Stability – the stability, or reliability, of a system is a key attribute for good performance. The system must be robust and dependable, and continue to provide service even if many users access it simultaneously or it experiences heavy usage over an extended period of time;
- Responsiveness – the perceived response time is the time from the beginning of the user action that generates a request to the end of the receipt of the response. The actual response time includes both network transmission and request-processing times;
- Scalability – the ability of a system to process an increasing load of requests without experiencing disproportionate increases in response times. In numerical terms a system is considered scalable if a graph of response time to number of users increases in a near-linear fashion;

*legassie@raytheon.com, mark@ipac.caltech.edu; phone 1 626 395-8688

Observatory Operations: Strategies, Processes, and Systems II, edited by Roger J. Brissenden, David R. Silva, Proc. of SPIE Vol. 7016, 70161C, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.788045



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

- Throughput – the amount of work (number of requests) processed by the system in a specified amount of time. Throughput analysis is used for performance tuning. The objective is to increase the amount of work the application can do.

The three most common types of load tests are stress tests, stability tests, and isolation tests. A stress test is designed to determine how heavy a load the system can handle. A huge load is generated as quickly as possible, and the time between transactions is minimized in order to intensify the load on the system. A stress test helps determine the maximum number of requests a Web application can handle in a specific period of time, and at what point the system will overload and break down (if at all).

A stability test is designed to determine whether a system will remain serviceable over an extended length of time. A considerable load is placed on the system for a long period of time, usually between 24 hours and a week. The load applied is designed to mimic the real traffic a system will encounter. Stability tests help determine whether the resource consumption of the system will remain stable throughout that time frame; whether the response times will remain low over an extended period of time; and whether there are technical faults such as memory leaks that may compromise the stability of the system.

Once a sequence of load tests has established that a problem exists in the system, an isolation test helps identify what the particular problem is. In an isolation test, specific sets of transactions are often executed over and over again in exactly the same way. The workload is designed to contain only the subset of test cases that caused the problem in previous testing. This type of testing helps pinpoint which set of requests caused a server to send back an error message or caused a deadlock on the database server.

In end-to-end testing normally a combination of stability and stress tests are performed to cover a wider range of system performance characteristics.

### 1.2 Benefits and Drawbacks

Load testing is a powerful and cost-effective technique for analyzing system performance. Proactive load testing during the design phase is a highly effective way for designers to minimize the time needed to bring a reliable system online, as well as an efficient way to detect weaknesses early in the development cycle. Once a computer system has been deployed, load testing is a very effective way to determine the impact new features and technologies, if added, would have on system performance. Load testing is also widely used as a powerful method for the ongoing analysis and tuning of existing systems.

Nevertheless, there are many ways in which load testing can deliver inaccurate or misleading analyses. Even the most carefully constructed load-testing environment cannot reproduce the real world perfectly. Some of the limitations include[1]:

- User simulation – there is no way to make virtual user behavior and actions perfectly mimic real users and the infinite variety of human behavior;
- Network imitation – load tests are usually executed on the local area network, whereas real-world traffic is sent over a wide area network and can encounter many kinds of problems such as network latency;
- Browser emulation – browsers use complex technologies like multi-threading and document caching which makes it difficult to emulate them accurately;
- Measurements – when monitoring a system care must be taken not to slow down the application and so influence its behavior.

Even beyond the technical limitations above, the creation of meaningful load tests is a complex and time-consuming task. There are many logistical problems that can compromise the load-testing process, including time-to-release deadlines or the lack of human or physical resources.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

# 2. BASICS OF LOAD TESTING

## 2.1 Load Testing Strategies

Load tests are necessary at every stage in the life cycle of the application, including design, development and operations. During development, performance analyses are needed when choosing among alternative designs. During operational use, load testing helps determine whether, or when, the system needs to be fine-tuned or even upgraded before more serious problems occur.

A load test should emulate the different types of traffic expected, in the volumes expected, using actual computer hardware and software.

Determining the correct strategy for load testing depends on various factors, most important being the architecture of the computer system under test. Most e-commerce applications and science archives are multi-tier systems organized into a number of components, each of which is located on a different server. The first tier, the client application (browser or standalone program), is known as the "front end" because the user directly accesses it. The other tiers, called the "back end", include a web server, application server, and database server. The basic communication protocol for data transfer on the internet is the Transmission Control Protocol/Internet Protocol (TCP/IP), and most applications now use another protocol that runs on top of TCP/IP called Hypertext Transfer Protocol (HTTP), used to establish communication rules for web applications and browsers.

Once the architecture and communication protocols are known, the first step is to determine what tool to use to measure its performance. Most load-test strategies for complex computer systems involve the use of an automated software tool since it's logistically difficult (if not impossible) to use human testers to send simultaneous requests to a system. Unfortunately, several years ago very few load-testing tools were available and many projects created their own rudimentary load-testing scripts and programs. These programs had limited capabilities and were often difficult to use. In recent years, however, several excellent load and performance test tools became available on the market, some of which are at no cost (freeware)!

Another important factor in determine the best strategy for load testing involves the type of computer application itself. For example, the load on a science data archive is much different than for a popular e-commerce website. Science archives typically have few users but large volumes of data downloaded by each user, whereas commercial websites tend to have many more users but less traffic and data processing.

## 2.2 Tool Selection

A tool feasibility study should be completed to determine which load-test tool will work best with your application. After defining the problem and analyzing needs, a list of required features should be generated, such as:

- Number of virtual users required
- Ease of use with minimal learning curve
- Support for multiple protocols and computing environments
- Emulation of various connection speeds (ADSL, WAP, cable modem)
- Support for CORBA, EJB, and/or JOLT
- Technology to emulate browser's document cache
- Automatic load distribution
- Integrated server monitoring
- Predefined reports and graphs to allow easy and comprehensive results analysis

Based on these requirements the next step is to define a list of available tools by researching the worldwide web, publications, exhibitions and/or conferences. This long list is narrowed down to a few tools using various pre-defined constraints such as environment (supports all your target environments), supplier competence, cost, and product quality. From this short list evaluations and demos are conducted during a trial period to choose the best tool for the system under test.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Information about the following small subset of load-testing tools can be found on the internet[2]:

- SilkPerformer – enterprise-class tool from Borland (formerly Segue). This powerful load-test tool can simulate thousands of users with multiple protocols and computing environments. The Spitzer Science Center (SSC) as well as Caltech's Infrared Processing and Analysis Center (IPAC) have used this tool for several years to load-test its operational data archives.
- WebLoad – load-testing tool from Radview Software. Basic version available as open source, add-ons available for purchase. Test scripting via visual tool or Javascript.
- Loadrunner – Hewlett Packard's (formerly Mercury's) load-testing tool for web and other applications. Direct competitor with SilkPerformer. Supports a wide variety of application environments, platforms and databases. Large suite of monitors to enable many performance measurements.
- IBM Rational Performance Tester – load-testing tool from IBM/Rational. Supports Windows and Linux as distributed controller agents.
- JCrawler – open-source stress-testing tool for web applications. User can give JCrawler a set of URLs and it will begin crawling from that point forwards, going though any URLs it can find on its way and generating load on the web application.
- e-Load – load-test tool from Empirix Software. Allows on-the-fly changes and has real-time reporting capabilities. Works with a wide variety of platforms. Oracle has recently acquired Empirix's test suite product line and is planning on incorporating this tool into their Oracle Enterprise Manager and Oracle Real Application Testing products.

At the Spitzer Science Center (SSC), two load-test tools were chosen for evaluation: SilkPerformer and Loadrunner. Both of these tools are powerful, enterprise-class applications able to satisfy the SSC's two most important criteria: 1) handle standalone applications such as SSC's Java-based client software (Spot & Leopard) used for science planning, proposal submission, and data archive access; and 2) support a multi-tier internet architecture system that utilizes Secure Socket Layer (SSL) protocols. The SSC decided upon SilkPerformer at the conclusion of the evaluation due to its ease of setup and use, as well as better technical support. Of note is SilkPerformer's intuitive Workflow Wizard, which guides the user step-by-step from project set-up to recording, testing, and analysis.

### 2.3 Test Scripts

In order to simulate real users as accurately as possible, the traffic between an actual user and the server is captured and recorded by the load-testing tool. At the SSC, Spot and Leopard applications use the HTTP protocol to send and receive serialized Java objects in binary format to/from a Weblogic application server. To capture this traffic, SilkPerformer and other tools use application hooking, a technique which allows a load-testing tool to hook into and record all Internet communications for the application under test. Another technique available for web browsers is proxy recording, which uses an intermediary (proxy) server to capture and record the traffic. Usually application hooking is easier than proxy recording since the load-test tool takes care of it automatically and the user does not have to modify proxy settings in the web browser.

After the traffic between a user application such as Leopard has been captured and recorded, it can then be used to run load tests. To create useful load tests, the recorded traffic must be modified in scripts to emulate real-world variations and different types of requests. A load-testing script is a specific set of operations for a virtual user to perform, designed to replicate real-world user activity. The script is written in scripting language, which generally speaking is a language based on a computer programming language and complemented with extensions for load testing. SilkPerformer uses a proprietary scripting language called Benchmark Definition Language (BDL), which allows for very advanced customization using standard programming techniques.

A load-testing script has certain essential characteristics: it contains transactions that describe the work, function calls to easily communicate with an application, and random variables to vary the behavior of a virtual user. The benefit of a script is that the tester can customize it in many different ways in order to replicate the varied patters of real-life traffic.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

### 2.4 Conducting Load Tests

The conduct of a load test involves the following eight basic steps:

1. Define the project (test plan) – test type (stress/stability/isolation), functions to be performed, typical user types, benchmarks and requirements
2. Determine workload model – increasing number of users (increasing workload), same number of users (steady-state workload), spreading total users evenly over time (queuing workload), or varying number of users (all-day or dynamic workload)
3. Record the script – record activities for typical users by launching the application via the load-testing tool. For SSC, SilkPerformer is used to launch Spot or Leopard and then automatically records the traffic between the user and SSC's web application server.
4. Customize the script – randomize data and paths of execution, handle cookies, assign transactions to user types, parse dynamic session information, and insert logic (e.g. loops). SSC uses extensive customization when load-testing its science data archive, such as replacing static conversation ids with variables and inserting a loop to check is archive packaging is complete.
5. Test the script – compile the test script and execute for a single user group, debug the script and update as necessary. SilkPerformer uses a built-in tool called True Log Explorer to easily validate the script is working properly.
6. Determine performance baseline – run a minimum load (one user) to determine the benchmark measurement, used for comparison during results analysis
7. Adjust workload and run test – select the workload model (usually steady or increasing), choose the scripts and user groups to run, specify time parameters including test duration, and the monitor the results during execution. SilkPerformer contains a monitor window that displays real-time execution status, as well as an output pane that displays data being transmitted and received, user information, timers, and errors.
8. Explore results – review log files, errors generated, response time measurements and other test output to look for performance problems (e.g. bottlenecks, server errors, crashes). SilkPerformer uses its Performance Explorer Tool to view an overview report and create custom graphs.

The SSC uses SilkPerformer to measure and record the response time (client-side measures) and throughput (server-side measures). Basic measures of performance and throughput recorded are requests per second received by the server, transactions per second system completed, total data volume sent and received, round-trip time, server busy time and number of concurrent connections. The SSC uses various server-monitoring tools, including SilkPerformer's Performance Monitor application, to monitor and record the archive server's CPU, memory utilization, disk activity, and network traffic. Finally, the SSC adds several custom timers when load-testing its science data archive with Leopard, including login time, query time, packaging time and download time. The use of these measurements can help pinpoint bottlenecks, uncover memory leaks, and provide information about different components of the server infrastructure.

## 3. ADVANCED TECHNIQUES

### 3.1 Case Study: Spitzer Science Center (SSC) Archive

The Spitzer Space Telescope is the fourth and final of NASA's great observatories. The observatory is designed to observe and explore the universe in the infrared at wavelengths ranging from 3.6 to 160 microns. It was successfully launched from Cape Canaveral in August 2003 and is performing exceedingly well[3]. Each year, the spacecraft returns thousands of images back to Earth where they are processed and archived at the Spitzer Science Center (SSC). The California Institute of Technology is the home for the SSC, which is responsible for the proposal process, user support, mission planning and scheduling, science data processing, and storage of archival quality data products. As of March 2008 approximately 14.4 Terabytes (TB) of data were stored in the archive, and by the end of the mission the archive is expected to grow to approximately 44 TB.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

A software application called Leopard is the primary interface to the SSC data archive. Leopard is a Java-based client-server software package that runs on a variety of platforms. A user can install Leopard on their own computer and use it to query specific observations from the Spitzer Archive, select various data products, and choose from direct data delivery to local disk or staged FTP. The basic downloadable products are derived from actual Spitzer observations called Astronomical Observation Requests (AORs). An AOR is the primary unit of Spitzer observing and consists of instrument parameters and astronomical target information. Each user normally requests tens, or even hundreds, of AORs to query and download. Each AOR can be up to several hundred Megabytes in size.

To load test the Leopard application, the SSC obtained the SilkPerformer load-testing tool with a configuration that allows up to 250 virtual users during run-time. Using the SilkPerformer load-testing tool, SSC captures the traffic between Leopard and SSC's server for later playback. Several different scripts can be created and played back simultaneously, such as scripts for querying data in various ways and scripts for downloading data.

The main Leopard window is shown in Fig 1 below.

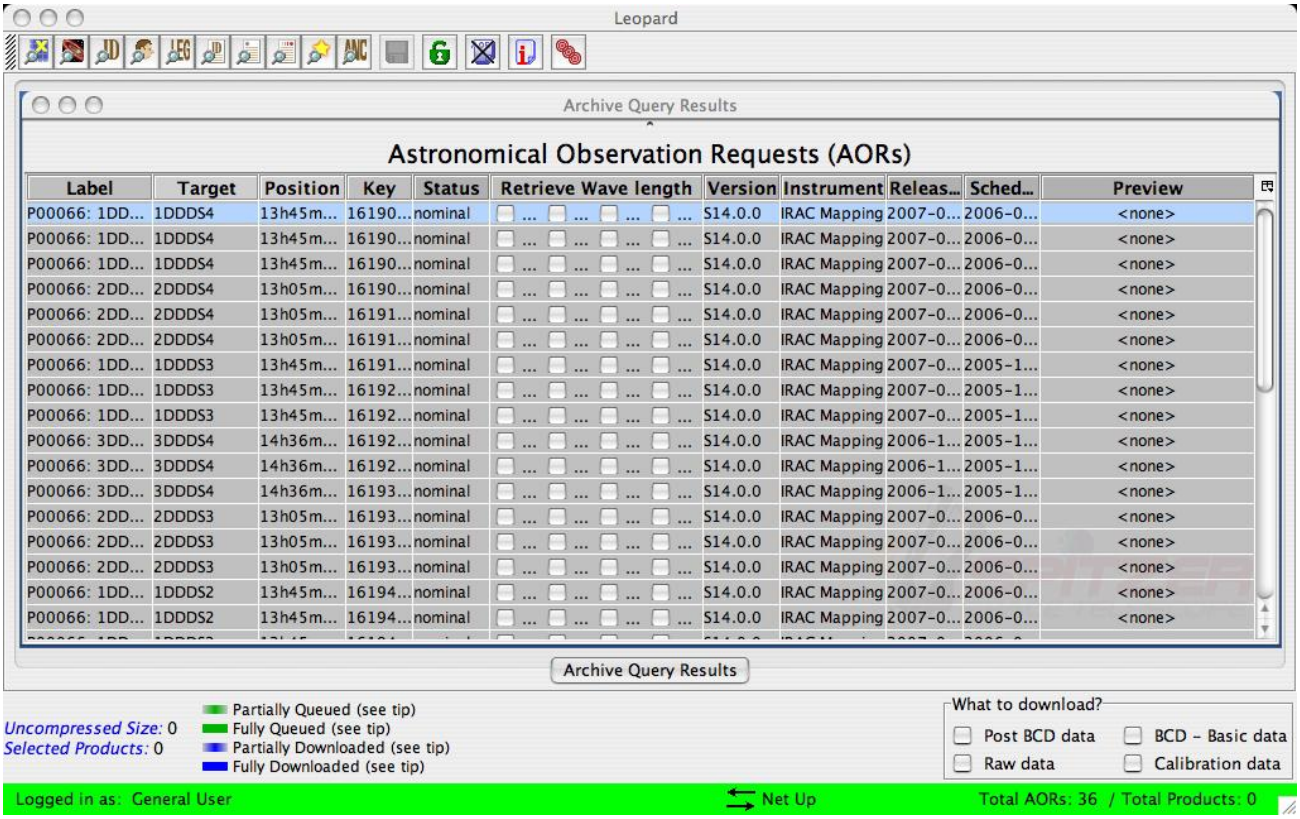


Fig. 1 The main screen of the Leopard application displays a list of Astronomical Observation Requests (AORs) queried from the Spitzer data archive.

The basic infrastructure of SilkPerformer includes four components:

- Controller – used for script development and load test management. The controller is the main component of the software. At the SSC the controller runs on a Windows-based computer connected to the local area network.
- Agent – executes the load test scripts, drives the load, and collects measurements. To handle 250 users, the SSC uses four agents running on four separate Windows-based computers. Agents are controlled by the Controller and at the SSC reside on the local area network.
- Results Repository – stores test results. Can be based on any of the following database systems: MSDE, SQL Server, Oracle, or DB2.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

- License Server – allows access to SilkPerformer and dictates characteristics such as maximum number of virtual users. At the SSC the license server runs on a separate Windows-based computer on the local area network.

SilkPerformer's main window showing a sample test script is provided in Fig. 2 below:

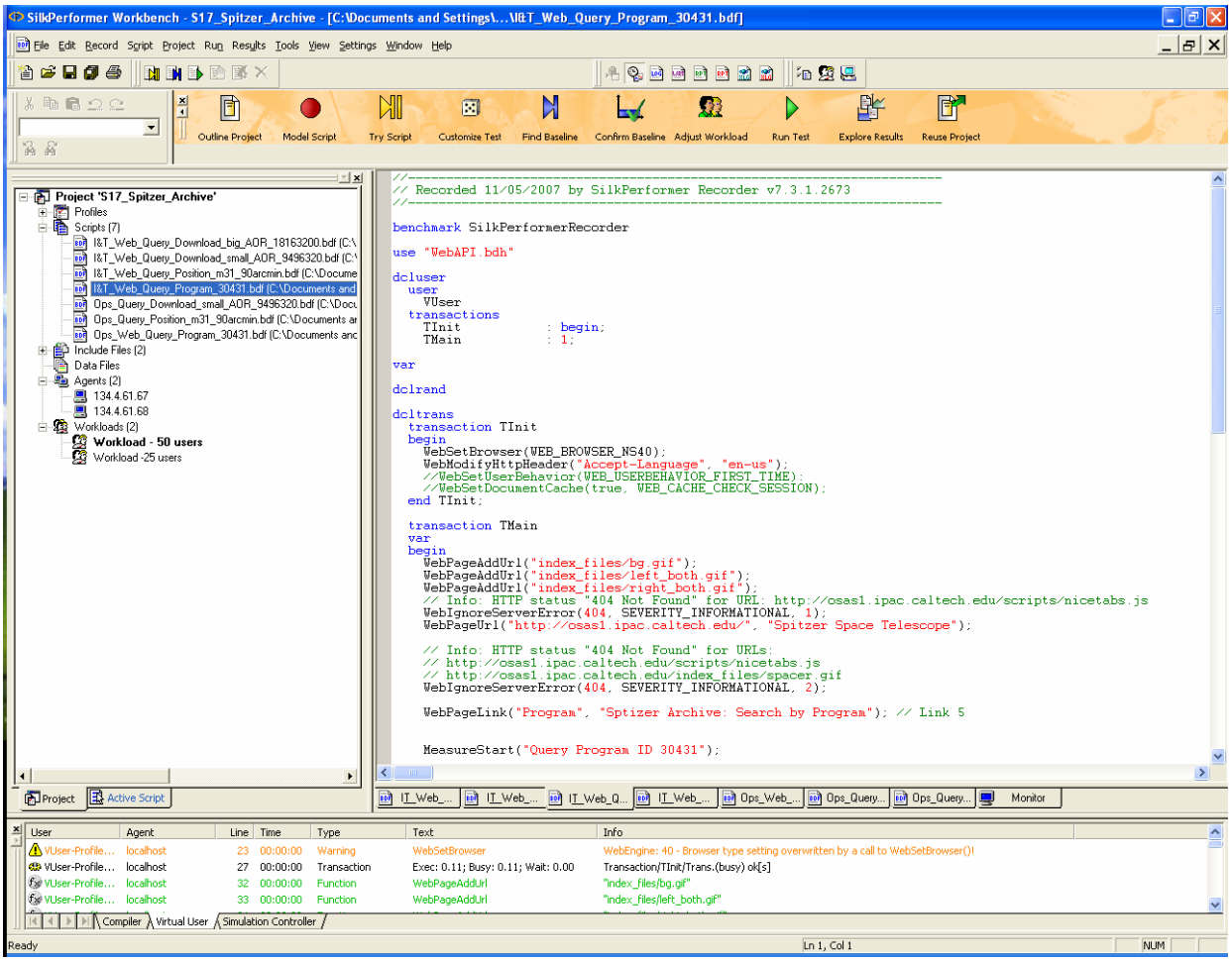


Fig. 2 SilkPerformer Main Window with test script to query the Spitzer archive

The strategy that SSC uses for the Spitzer archive combines two different types of load tests: stability and stress. For both of these tests groups of different users perform up to 5 different functions, including querying by position (RA/Dec), querying by program, downloading by AOR ID, and AOR visualization (a feature which allows images of the target region to be downloaded and displayed).

To determine how many virtual users to use as well how to load the system in a realistic fashion, SSC used data from other similar missions and interviewed the scientific user community planning on using Spitzer. In addition, a requirement to handle 150 users simultaneously was imposed on the SSC. Of note, this requirement ended up being too vague and not realistic, since the processing demands for user requests vary drastically, and SSC is limited on the number of allowable concurrent server connections (currently 30) due to resource limitations. This means that if 150 users are attempting to access the archive simultaneously, only 30 will be successful. The remaining 120 will be placed in a waiting queue. For SSC's data archive it was decided that 25 simultaneous users querying and/or downloading data is more than sufficient to emulate the maximum realistic load. As stated earlier, a science data archive such as Spitzer's



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

typically has much fewer users than a commercial website since it targets only the astronomical community. However, the volume of data downloaded by each user is an order of magnitude larger than the typical commercial website. Therefore, SSC's load tests are specifically designed to measure the time-consuming packaging process, which retrieves the data products and compresses them into a zip file before downloading.

Over the past 3 years of operation users have downloaded on average about 500 Gigabytes of data using Leopard each week, with a few spikes up to 2500 GB in late 2006 (see Fig 3. below). These anomalous spikes were caused by an application interface used by another infrared archive and have since been eliminated.

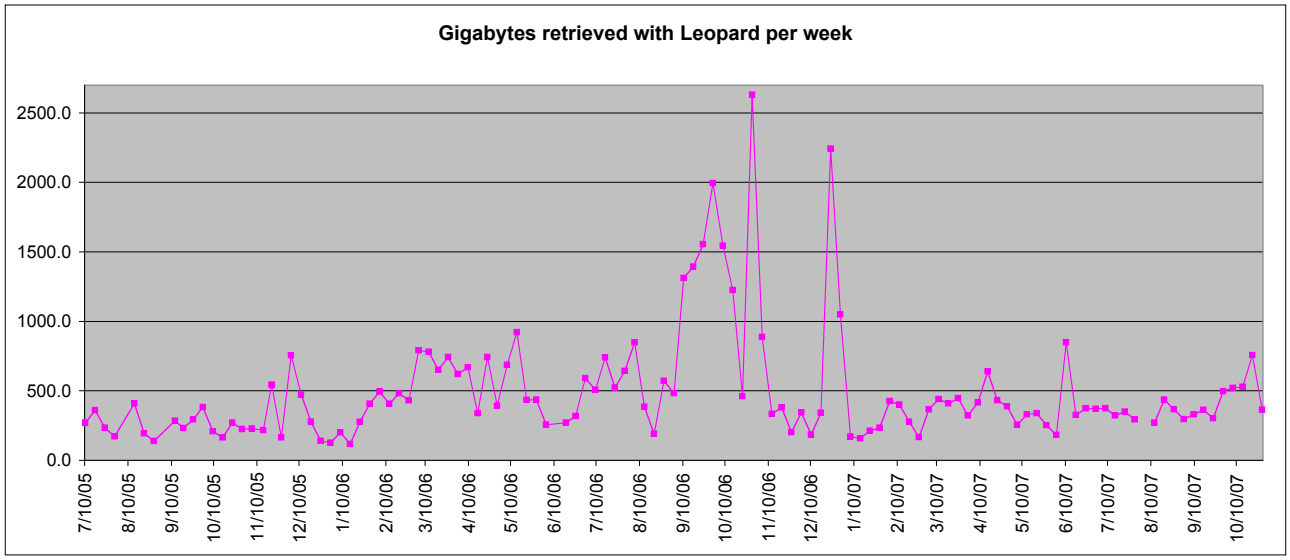


Fig. 3 Gigabytes downloaded by users from the Spitzer archive (Jul 2005 – October 2007)

For stability tests, SSC normally runs two load tests on the archive using a total of 25 users and 50 users, respectively. The users are split into 5 user groups performing different functions such as querying and downloading. Each load test runs up to 10 hours. An example of a load test that was run in December 2007 is shown in Fig 4 below.

| TRANSACTION | # Transactions (25-user test) | # Transactions (50-user test) |
|---|---|---|
| Query Only (by Position) M31 90Õ | 259 | 1769 |
| Query Only (by Program) 30431 | 679 | 3689 |
| Part 1 - Query AOR 9496320 | 4932 | 655 |
| Part 2 - Package AOR 9496320 | 4917 | 625 |

Fig. 4 Number of transactions processed during Spitzer archive load test for 25-user and 50-user tests

Note that the number of packaged AORs dropped by an order of magnitude (from 4917 to 625) when increasing from 25 to 50 users, even though the 50-user test lasted 3 times longer. This could indicate a problem with load balancing or server fine-tuning.

For stress testing, SSC typically uses an increasing workload, starting with one user and adding an additional user every 5-10 minutes up to a maximum of 100 users. Normally errors begin around 50 users – typical errors reported to users include HTTP 1062 (HTML form not found) and HTTP 500 (internal server error) issues.

As a result of stability and stress testing of the Spitzer science archive, bottlenecks were uncovered early in the design phase which resulted in the increase in the number of allowed concurrent connections as well as a change in the number of the packaging servers deployed.

### 3.2 Resolving Runtime Issues

Several issues can occur during runtime that do not occur during the recording process. For the SSC data archive, which is a large and complex multi-tier system, the following problems were uncovered:



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

- Conversation ID – Conversation IDs are assigned by SSC's web application server and are unique for each request. During recording SilkPerformer records the actual converation ID for that particular request, but since it changes each time, future attempts to run the script fail.
- Product Filename – Leopard provides a unique path and filename for each download request so that Leopard knows where to retrieve the data products. SilkPerformer records the static filename, but since it changes each time, future attempts to run the script fail.
- Download Status – Leopard uses a separate program called Subscriber to track the packaging (staging) process and inform the user when the products have completed staging and are ready for download. SilkPerformer is unable to hook into Subscriber and cannot track the status without adding extra logic to the test script.

To resolve these three runtime issues, SSC utilized SilkPerformer's powerful customization functions as well as manually adding new logic into the test scripts. To address the conversation ID problem, SSC changed the static HTTP conversation ID to a variable. New logic was inserted into the script to build a unique conversation ID using the current time stamp, IP address and virtual user number. See Fig. 5 below for the actual test script modification showing the replacement of the static ID in the binary code with a variable called sConversationId.

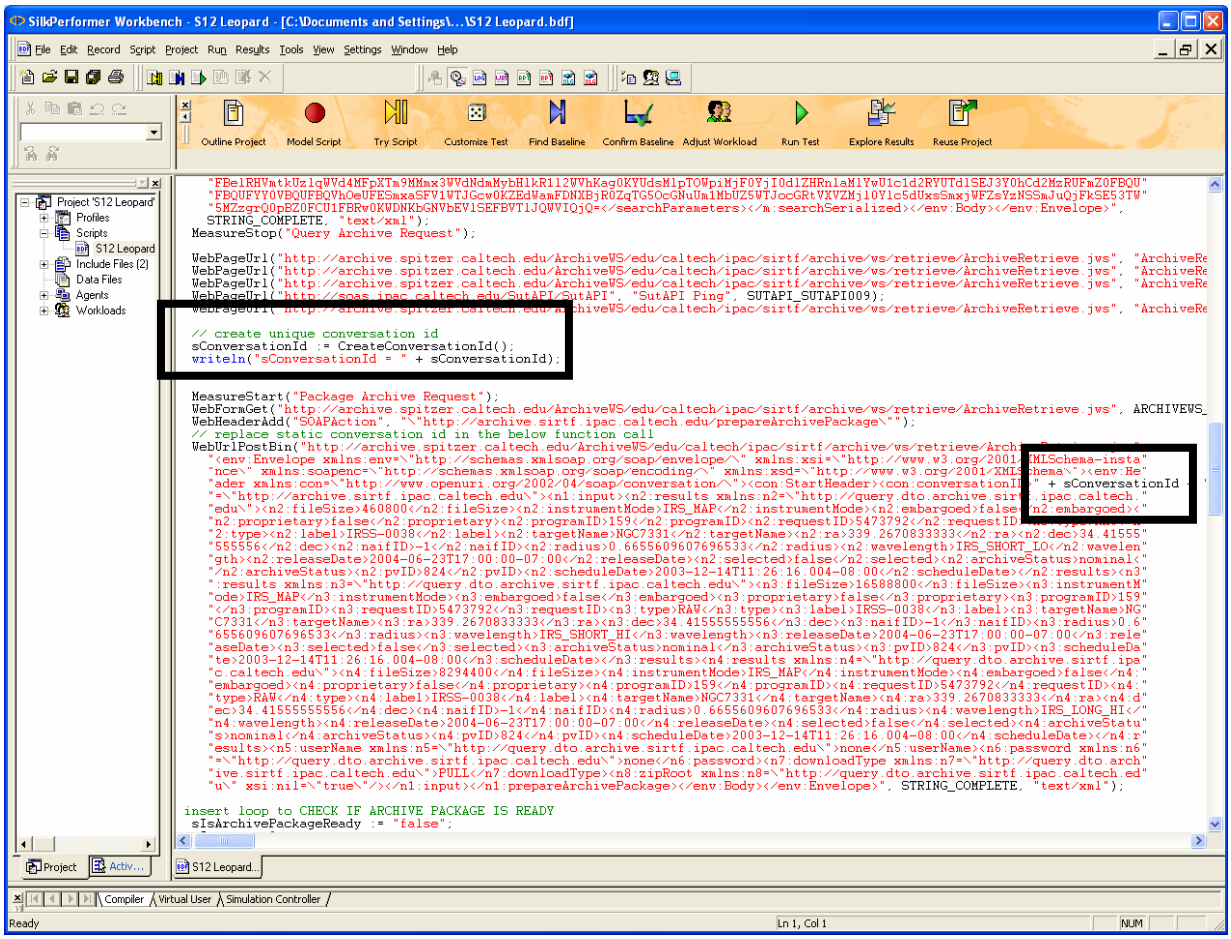


Fig. 5 Load test script showing the manual replacement of the static conversation ID with a variable

The variable sConversationId, which contains the unique conversation ID, is inserted into the test script and replaces the static ID. During runtime, SilkPerformer automatically calculates new conversation IDs and inserts them into the request before sending it to the SSC server. This complex process is simplified greatly thanks to SilkPerformer's ability to insert basic logical operators and functions (including variables) directly inside the serialized Java package.



Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

To resolve the product filename issue, SSC utilized a technique similar to the conversation ID. Since the server creates a unique path name for staging and retrieval, logic was added to parse the response from the server informing the user that staging is complete. This parsing activity reveals the actual filename, which is then inserted into the download request before sending it to the server.

The final problem remaining was the download status checking, which SilkPerformer was unable to record since a separate Java application (Subscriber) spurred off by Leopard was involved. Without this logic, SilkPerformer did not know when packaging was complete and therefore could not measure response time or downloaded any file. To resolve this issue, SSC inserted a loop to check if the archive packaging process was complete. To make the test as realistic as possible, the logic replicated that exact behavior of the original Subscriber code. For example, on the first attempt SilkPerformer waits for 20 seconds before sending a request to the server requesting server status. After six attempts, Leopard only checks every 45 seconds if packaging is complete. After 12 attempts, Leopard waits 150 seconds before sending a status request.

In summary, the SSC has been able to utilize advanced methods and an enterprise-class load-testing tool to effectively and efficiently measure the performance of its data archive. Advanced techniques such as those mentioned in this paper are becoming more necessary as new data dissemination technologies emerge and sophisticated client-server architectures mature.

## 4. CONCLUSIONS

Load testing is a powerful and effective method for ensuring that an application can handle realistic loads prior to release. In addition, load testing is the only way system performance, stability, and reliability can be evaluated under actual conditions of use. Several load-testing tools are available, from free open source software to enterprise-class applications. SSC discovered through evaluations and analyses that the SilkPerformer load-testing tool from Borland (formerly Segue) fit all requirements for testing NASA's Spitzer Space Telescope science data archive. Advanced load-testing techniques were used by the SSC to overcome several runtime issues. These techniques involved modifying the load-test scripts by adding variables and logic to create unique and realistic load-test scenarios. As a result of load testing, SSC has been able to fine-tune Spitzer's data archive to perform more efficiently and with much fewer problems. In addition, bottlenecks were uncovered early in the design phase which resulted in the increase in the number of allowed concurrent connections as well as a change in the number of the packaging servers deployed.

## REFERENCES


[1] Asbock, S., [Load Testing for eConfidence], Segue Software, Lexington (2000)

[2] Hower, R., Software QA/Test Resource Center, "Web Site Test Tools and Site Management Tools", www.softwareqatest.com (2008)

[3] Bennett, L., Comeau, S., Levine, D., and Dodd, S., "The Spitzer Science Center System Description and Lessons Learned Due to Two Years of Operations", Proc. SPIE 6270 (2006).




Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 7/19/2018
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use